DirectCertify

Unlock Your Potential, Certify Your Success

Snowflake

NAS-C01 Exam

SnowPro Specialty - Native Apps

Exam Latest Version: 6.0

DEMO Version

Full Version Features:

• 90 Days Free Updates

• 30 Days Money Back Guarantee

• Instant Download Once Purchased

• 24 Hours Live Chat Support

Full version is available at link below with affordable price.

https://www.directcertify.com/snowflake/nas-c01

Question 1. (Multi Select)

You are designing a Snowflake Native App that needs to store temporary state information specific to each consumer account. This state information should not be directly accessible to the consumer but must be persistent across application sessions. Which of the following approaches are valid and recommended for storing this data?

A: Create a public schema in the application package and store the state information in tables within that schema. Grant SELECT access to the consumer's account on these tables.
B: Utilize internal stages within the application package to store the state data as files.
C: Create secured views that expose the state data but control access through application roles.
D: Store the state information in a private schema within the application package, ensuring that consumers do not have direct access.
E: Use external stages connected to a provider controlled AWS S3 bucket or similar.

Correct Answer: B, D

Explanation:

Options B and D are the correct approaches. Option D: Private schema, Consumers should not have direct access to the app's internal data. Storing state information in a private schema within the application package ensures that consumers cannot directly query or modify this data, preserving the application's internal state and security. Option B: Internal Stages, offer a way to store file-based data that is tightly coupled with the application package. It enables you to persist the state information for each consumer securely. Option A is incorrect as it gives consumers direct access. Option C is incorrect because the requirement says the data cannot be accessible, not that it should be controlled. Option E creates dependency on external infrastructure.

Question 2. (Single Select)

You are developing a Snowflake Native App that allows consumers to enhance their existing customer data with enrichment data provided by your app. The app relies on secure data sharing. To minimize data duplication and maximize query performance within the consumer's

account, which of the following approaches offers the MOST optimized and secure way to deliver the enrichment data?

A: Create a staging table in the provider account, load the enrichment data into it, and grant the consumer account SELECT privileges on this staging table. The consumer then creates their own table and copies the data.
B: Utilize Snowflake Secure Data Sharing to directly share the enrichment data tables from the provider account to the consumer account. The consumer can then create views on the shared tables or query them directly.
C: Develop a Snowflake UDF (User-Defined Function) that, when called in the consumer account, retrieves the enrichment data from an external API endpoint (maintained in the provider account) and returns it.
D: Create a secure view within the application package itself, which joins the application's enrichment data with consumer-provided data via an API integration. The consumer queries this secure view.
E: Use Snowflake Data Marketplace listing to provide enrichment data.

Correct Answer: B

Explanation:

Option B (Utilize Snowflake Secure Data Sharing) is the most optimized and secure approach. Secure Data Sharing avoids data duplication, minimizes latency, and allows the consumer to directly access the enrichment data without needing to copy or move it. It leverages Snowflake's native sharing capabilities, ensuring security and governance. Option A involves data duplication. Option C introduces external API dependency and potential performance bottlenecks. Option D mixes consumer data with the application code, which is not optimal and less secure. Option E is suitable for data discoverability and monetization, not for application-specific, customized data enrichment.

Question 3. (Single Select)

You are developing a Snowflake Native App that will be listed on the Snowflake Marketplace. The application requires a specific version of a Python library that may conflict with other libraries used by consumers. How can you ensure that your application uses the correct library version without causing conflicts in the consumer's environment?

A: Instruct consumers to manually install the required Python library version in their Snowflake environment before installing your app.
B: Bundle the required Python library with your application package and utilize Anaconda packages support to manage dependencies, ensuring isolation from the consumer's environment.
C: Modify the consumer's 'PYTHON PATH' environment variable upon installation to include the location of your application's Python libraries.
D: Rely on Snowflake's automatic dependency resolution to identify and install the correct version of the Python library in the consumer's environment.
E: Package all application logic into a single SQL UDF to avoid Python dependency issues entirely.

Correct Answer: B

Explanation:

Bundling the required Python library within your application package and leveraging Anaconda is the recommended approach. This creates an isolated environment for your application's dependencies, preventing conflicts with other libraries in the consumer's environment. Manual installation by the consumer is inconvenient and prone to errors. Modifying the is not a supported or recommended practice. Snowflake's dependency resolution may not guarantee the correct version for all cases, especially with conflicts. Limiting to SQL UDFs might not be feasible or efficient for complex Python-based logic.

Question 4. (Single Select)

A Snowflake Native App is designed to perform complex data transformations on large datasets provided by the consumer. To optimize performance and manage resource consumption, you need to choose the appropriate execution context for your application logic. Which of the following options should be considered to accomplish this?

A: Executing all data transformations within a single stored procedure to minimize overhead.
B: Distributing the transformations across multiple SQL UDFs running in parallel using Snowflake's compute grid.
C: Leveraging Snowflake's external functions to offload data transformations to a separate cloud-based compute environment.
D: Using a combination of stored procedures, SQL UDFs, and external functions to orchestrate

the data transformation process, depending on the specific requirements of each step.
E: Run entire process by calling API from provider side.

Correct Answer: D

Explanation:

Choosing the correct execution context for your application logic requires a combination of stored procedures, SQL UDFs, and external functions to orchestrate the data transformation process, depending on the specific requirements of each step, offering the best balance of performance, scalability, and resource management. A single stored procedure may not be efficient for complex transformations. SQL UDFs in parallel can be effective but might have limitations depending on the complexity. External functions offer flexibility for offloading but introduce latency and complexity. API call will be more appropriate to transfer the data.

Question 5. (Multi Select)

You are developing a Snowflake Native App that needs to persist state information between different invocations. The app requires tracking of user preferences, processing progress, and other runtime dat

a. Which of the following options are viable and secure methods for persisting this type of state information within the context of a Snowflake Native App?

A: Using temporary tables within the consumer's account to store state information. The tables are dropped when the app is uninstalled.
B: Using the application provider's own Snowflake account to store state information associated with each consumer.
C: Storing state information within the application package itself by updating the package version with each state change.
D: Using internal stages and secured views in consumer account to persist state information.
E: Using secure external stages managed by the application provider to store the consumer-specific state information.

Correct Answer: B, D

Explanation:

Persisting state information requires careful consideration of security, scalability, and isolation. Using the application provider's own Snowflake account is a viable approach if designed carefully, but requires robust security measures to isolate consumer data. Internal stages and secure views offer a more secure and controlled way to persist state within the consumer's account. Temporary tables are ephemeral and unsuitable for long-term state persistence. Storing state within the application package is impractical and would lead to versioning issues. External stages managed by the provider introduce complexity and potential security risks.

DirectCertify

Unlock Your Potential, Certify Your Success

Full version is available at link below with affordable price.

https://www.directcertify.com/snowflake/nas-c01

30% Discount Coupon Code: LimitedTime2025